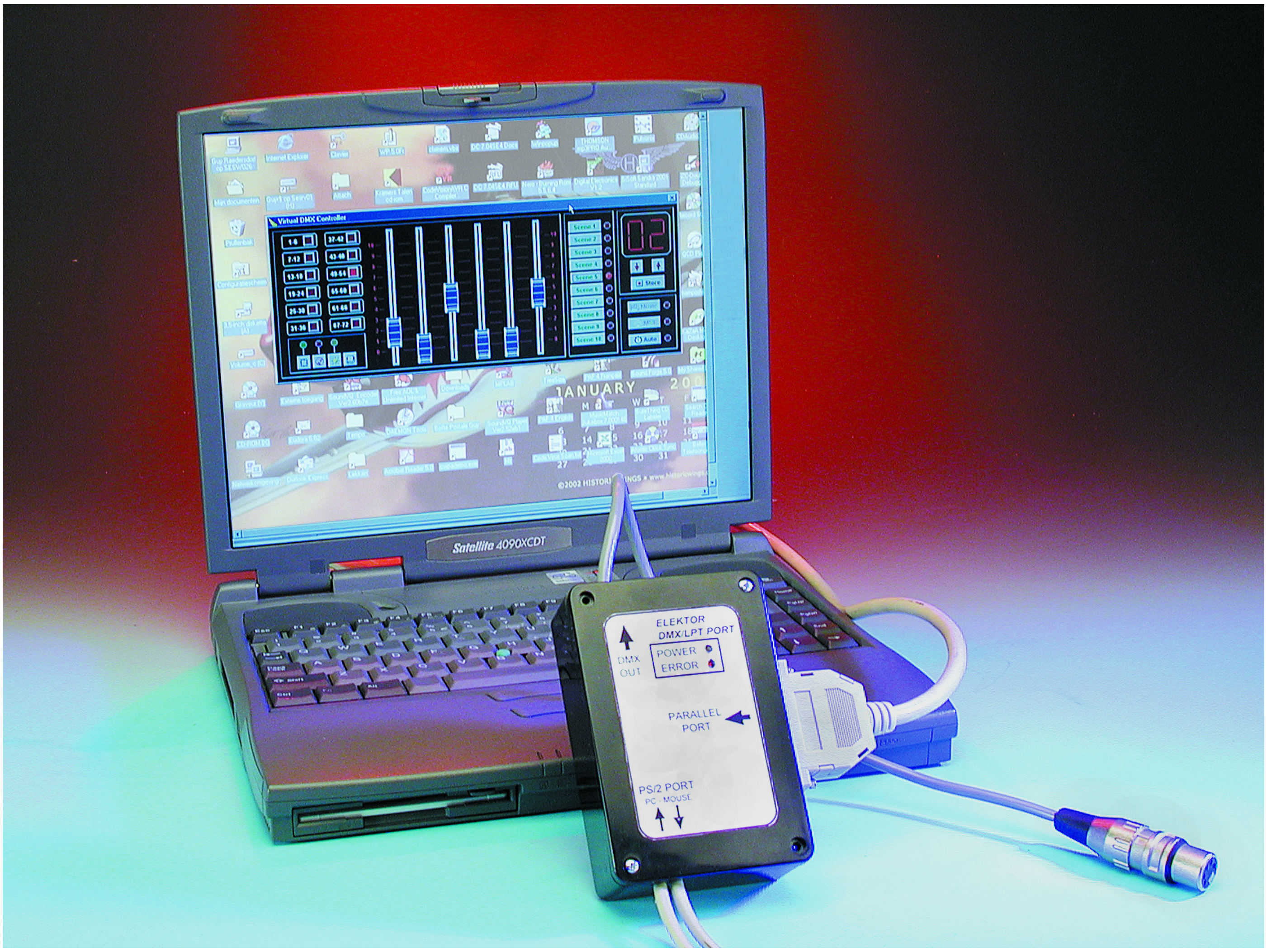
LPT/DMX Interface

480 DMX channels via the parallel port

Design: B. Bouchez

bbouchez@netcourrier.com

The DMX/MIDI interface described in the September 2001 issue is extremely flexible and suitable for nearly all applications, but it is a rather complex circuit. By contrast, the interface described here is the height of simplicity with regard to both construction and programming, even under Windows.



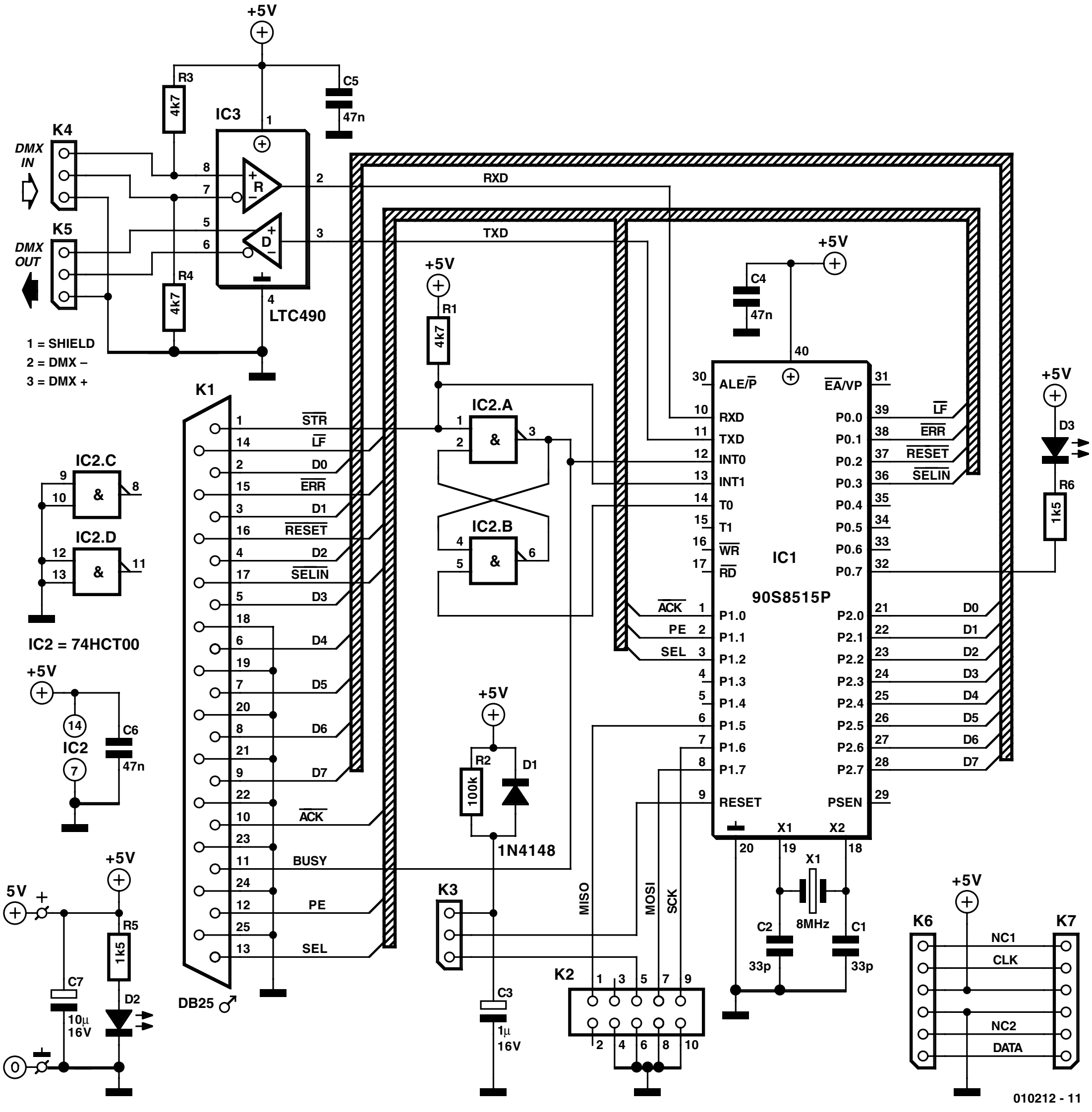


Figure 1. The electronics of the LPT/DMX interface are concentrated in an Atmel AT90S8515 IC.

The DMX protocol has already been the subject of several articles in *Elektor Electronics*. Consequently, here we will describe only the essential points in order to refresh your memory. Readers who are interested in the technical details of the protocol are referred to the article in which it is described ('DMX512 Revealed' in the May 2001 issue) and to various DMX designs that have subsequently appeared in *Elektor Electronics* ('MIDI/DMX Interface' and 'DMX Demultiplexer').

The DMX512 protocol

The protocol was defined in 1986 by the USITT. This American organisation is responsible for the standardisation of all technical matters in the 'show biz' area, such as exchanging digital information between control panels and devices such as dimmers and automated lighting equipment.

The DMX protocol is based on a unidirectional serial link meeting the specifications of the RS485 standard. The commands that are intended to be received by the 'slaves' are sent in the form of a series of bytes, each of which controls one DMX channel. The link works at a speed of 250 kbaud with the format '8 bits, no parity and 2 stop bits'. In order to allow the start of each data frame (which can be as large as 512 bytes plus the start byte) to be determined, communications are interrupted ('Break') for the duration of at least two bits, which allows the receivers to synchronise themselves.

Why this interface?

The DMX signal can be generated by a simple serial port (which, by the way, was also what the developers of this standard had in mind). However, the serial port of a PC is totally unsuitable for this, for two reasons. First, PC ports are designed to meet the RS232 standard (although an RS232/RS485 converter might provide a solution here), and second, the clock rate of serial PC ports makes it impossible to generate a 250-kbaud signal.

To make a long story short, if you want to use DMX in combination with a PC, a separate interface is necessary. The approach that has been taken here is basically very simple.

COMPONENTS LIST

Resistors:
R1,R3,R4 = 4kΩ7
R2 = 100kΩ
R5,R6 = 1kΩ5

Capacitors:
C1, C2 = 33pF
C3 = 1µF 16V radial
C4,C5,C6 = 47nF
C7 = 10µF 16V radial

Semiconductors:
D1 = 1N4148
D2 = LED, red, high efficiency
D3 = LED, yellow, high efficiency
IC1 = AT90S8515-8PC, programmed, order code **010212-41**
IC2 = 74HCT00
IC3 = LTC490 CN8 (Linear Technology)

Miscellaneous:
K1 =DB25 plug (male), PCB mount, angled pins
K2 = 10-way boxheader
K3,K4,K5 = 3-way SIL pinheader
K6 = 6-way mini-DIN socket, PCB mount
K7 = 6-way SIL pinheader
X1 = 8MHz quartz crystal
PCB, order code **010212-1** (see Readers Services page)
Disk, source code and programs, order code **010212-11**

Project software and PCB layout also available as free downloads from www.elektor-electronics.co.uk

We just use the parallel port, which is supported by most operating systems (DOS, Windows 3.x/95/98/ME etc. and LINUX). Furthermore, the PC is far from being the only type of computer with a parallel port. It is thus possible to use this interface with machines such as the (former) Atari, Amiga and so on, as long as the proper software is available.

In order to keep this interface as universal as possible and thus avoid limiting its use to modern PCs, it employs only the Centronics protocol instead of any of the PC-specific protocols (Nibble, Reverse, EPP and ECP).

This approach also has a significant advantage when the interface is used with Windows. With this operating system, it is normally necessary to develop drivers in order to be able to use specific interfaces, which is rather complicated. Since this interface uses the Centronics protocol, Windows simply considers it to be a printer and thus installs the standard drivers.

This means that even under Windows, programming the interface amounts to nothing more than ‘printing’ control commands. This interface can thus be controlled using any desired programming language (Delphi, Visual Basic, Visual C, C++ Builder etc.) that supports printing under Windows.

Furthermore, the designer of the interface has written a special Windows module that allows you to control it without having to know how to program the parallel port under Windows. There is also a special program for testing the interface. Finally, we should note that the interface is supported by the Soft Controller programs (more about this later on).

The hardware

As can be seen from **Figure 1**, the schematic diagram of the interface is rather simple, since IC1 (an Atmel AT90S8515 RISC microprocessor) looks after practically everything. It contains just about all that we need: RAM, a serial port and EEPROM for the program. A few years ago, around ten chips would have easily been necessary for such a circuit.

The only other semiconductors we need are IC2 (which is used to implement a flip-flop) and IC3 (a Linear Technology LTC490CN8 RS485 transceiver). By the way, the LTC490 may be replaced by its equivalent, the SN75179 (which may be easier to find).

In our first prototypes, the Strobe line of the parallel interface was connected directly to one of the microprocessor interrupt inputs. The width of the pulse on this line varied so much that sometimes things didn’t work properly. This is why the flip-flop formed by IC2a/b was added. With this change, even very short pulses are registered.

The pin assignments of K1 match those of the parallel port connector on the PC. The interface can thus be connected to the Centronics connector using a standard cable.

IC3 is a RS485 transceiver. It converts the TTL signals from the microprocessor’s serial port to the RS485 signals required by the DMX standard. To keep things simple, we have not provided electrical isolation between the microprocessor and the transceiver. If you are worried about excessive voltages due to earth loops, you should fit Transil protectors between the DMX+ and DMX– lines on K4 and K5 (pins 2 and 3).

You may have already noticed that the DMX interface has a DMX IN input, but this input is not used! It has been included for possible future extensions.

LED D2 indicates whether the interface has power. The job of LED D3, which is driven by the microprocessor, is to report any errors that may occur in the commands send by the computer.

Since this interface can be powered from the PC to which it is connected, the values of R4 and R5 are chosen for use with low-power, high-efficiency LEDs. If you decide to use standard LEDs, a value of 220 or 330 Ω is suitable for these resistors.

Thanks to its low current consumption (less than 10 mA), the interface can be powered directly from the PC to which it is connected. A voltage regulator is thus not necessary.

To obtain power from the PC, we make use of the connection from the PC to the keyboard, which is powered from the 5-V supply. Connectors K6 and K7 are connected to a simple keyboard cable (PS/2 or AT), and the necessary voltage can be tapped off from this cable.

Connector K2 is only used for programming the microcontroller. The Atmel programming cable can be fitted directly to this connector (see the Atmel site at www.atmel.com for the schematic diagram). If you order a pre-programmed microprocessor from Readers Services (order number **010212-41**), K2 may be omitted.

Finally, the triplet R2/C3/D1 generates the reset pulse for the microprocessor. This pulse runs via K3, where you can select the source for the reset (connector K2 or the reset circuit). It should be clear that the K3 jumper should not be fitted if you want to program the microprocessor yourself. During normal operation, the jumper must be present on K3 to provide a link to R2/C3. If you purchase a pre-programmed microprocessor, you can simply replace K3 with a wire bridge fitted between the connections for pins 2 and 3.

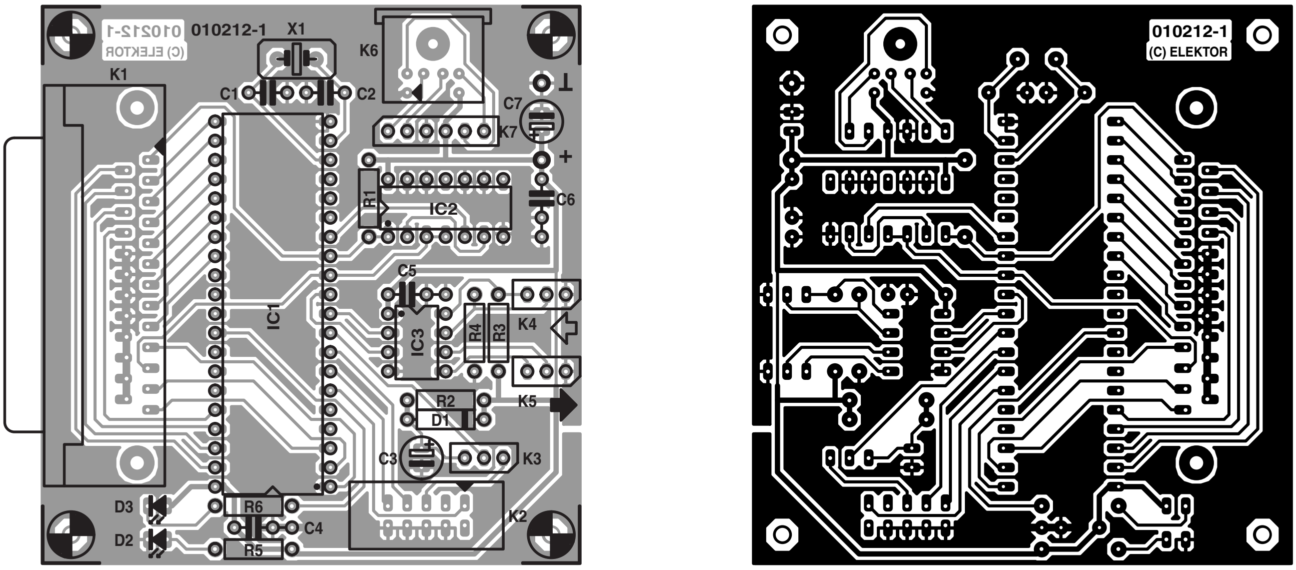
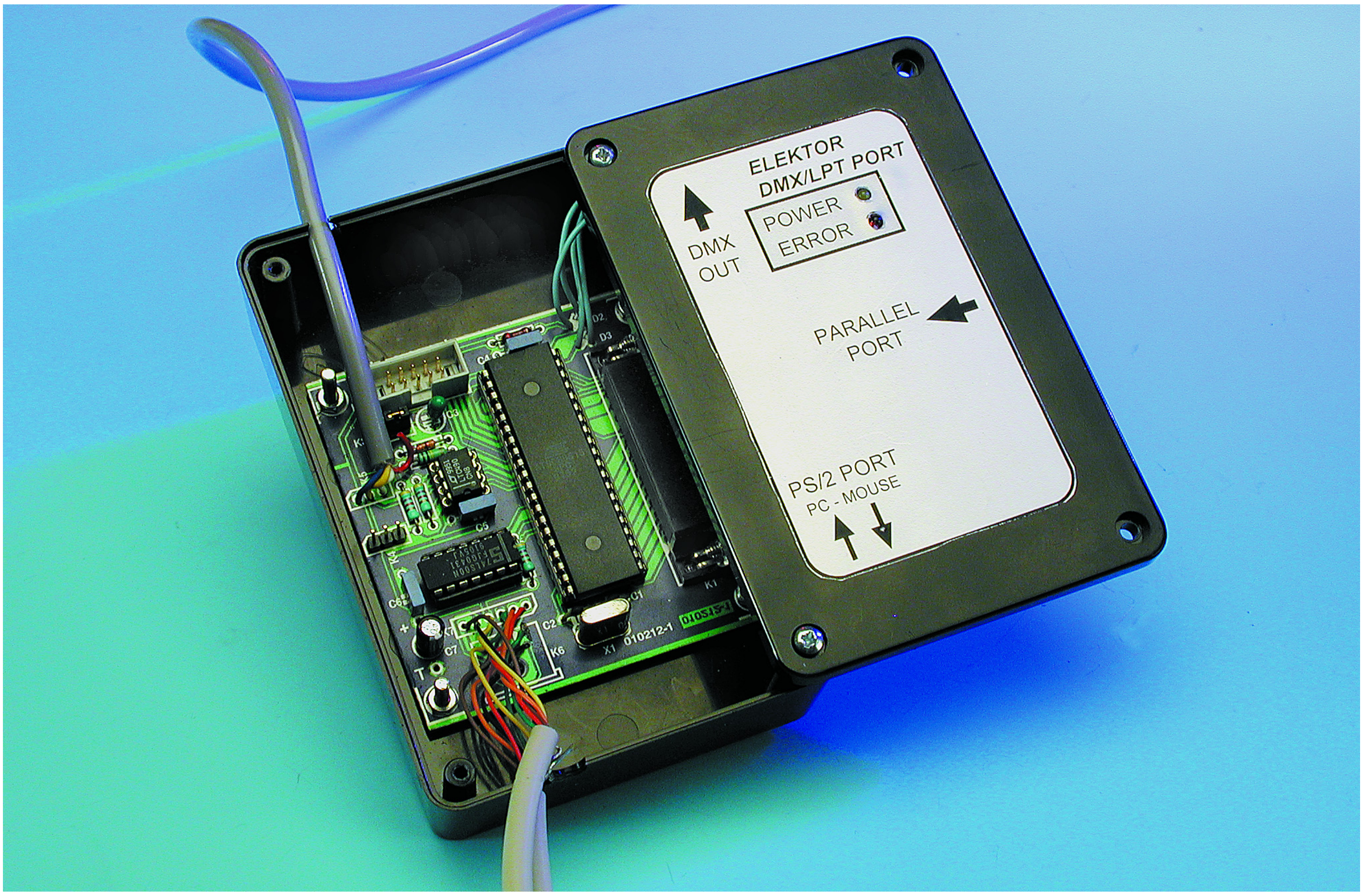


Figure 2. The design of the single-sided printed circuit board is quite simple, so there's not much that can go wrong during assembly.

Here comes the soldering iron

Thanks to the small number of components, assembling the interface is relatively simple. **Figure 2** shows the track and component layouts for the circuit board. In spite of its small dimensions, this is still a single-sided circuit board (the author has made a circuit board that was only half as large by using the PLCC version of the Atmel microprocessor). Use good-quality sockets for the three ICs. If you want to connect K1 of the interface directly to the DB25 connector on a PC, be sure to use a connector without integrated securing screws for K1, since otherwise the connector cannot be



plugged into the PC.

As far as the power supply is concerned, there are various options. The simplest solution is to draw power from the mouse or keyboard cable present on every PC. You can also power the interface from an external supply with a 5-V regulator, but that is more expensive.

If the interface is to be powered from the mouse or keyboard cable, the simplest solution is to take a DIN/DIN extension cable and cut it in half. The wires from the two halves of the cable can then be soldered to the proper pins of connectors K6 and K7. **Figure 3** shows the pin assignments of the PS/2 and DMX cable and chassis connectors.

On the circuit board, K6 is implemented as a PS/2 bus connector. If you fit such a connector to the circuit board, all you have to do is to solder a length of cable with a PS/2 or DIN plug to K7. The original mouse or keyboard plug can then be inserted into the PS/2 bus connector on the circuit board.

Table I: Interface commands.

Command ‘D’	Change a group of values for a series of DMX channels.
syntax:	ESC D hi lo nn dd dd dd...
parameters:	hi: high byte of the first DMX channel to be changed lo: low byte of the first DMX channel to be changed nn: number of DMX channels to be changed (1–255) dd: DMX value for the first channel to be changed dd: DMX value for the second channel to be changed dd: DMX value for the third channel to be changed etc. … The number of values (dd) must match the parameter nn.
Command ‘d’	Change the value of one DMX channel (number 1–256).
syntax:	ESC d nn dd
parameters:	nn: DMX channel number (0–255 for channels 1–256) dd: the value to be sent
Command ‘e’	Change the value of one DMX channel (number 257–480).
syntax:	ESC e nn dd
parameters:	nn: DMX channel number (0–224 for channels 257–480) dd: the value to be sent
Command ‘F’	Set the standard parameters (Start Code = 0 and Break Time = 100 µs).
syntax:	ESC F
Command ‘I’	Initialise the interface (also occurs when the power is switched on).
syntax:	ESC I
Command ‘S’	Change the Start Code
syntax:	ESC S ss
Command ‘Z’	Completely reset of the DMX memory (all channels set to 0).
syntax:	ESC Z
Command ‘T’	Change the Break Time setting.
syntax:	ESC T tt tt: Break Time setting (in steps of approximately 50 µs; standard value is ‘2’ for 100 µs)

The enclosure

The final subject is fitting the circuit into an enclosure. Given the small dimensions of the circuit board, that should not present any difficulties.

You can fit the circuit board into a separate enclosure that is connected to the parallel port of the PC by a suitable cable (see the photo at the head of the article), or you can fit the circuit board into a small enclosure that can be attached directly to the parallel port connector of the PC.

Bear in mind that the latter solution is not always the most convenient, due to the size of the circuit board. With an office computer, there is usually enough free room at the back, but with a laptop the other connectors are usually so close to the printer connector that your little DMX box will make it more or less impossible to use the other connectors.

The circuit board is not particularly sensitive to external noise, but it is still recommended to use a screened plastic enclosure (i.e., one with an internal graphite coating connected to ground).

The DMX output is implemented in an especially simple manner, namely using a short length of cable fitted with a 3-pin or 5-pin XLR connector. This cable is soldered to K5.

Centronics language

As soon as the circuit is connected, you can start testing and programming.

Thanks to the simple design, it is most likely that everything will work properly from the start. However, a special program has been written to make it easier to test the circuit board. This program is very easy to use, but nevertheless we have summarised the various commands in a separate file on the floppy disk.

Besides providing special functions for testing the interface (such as switching on Error LED D3), this program is also suitable for testing spotlights.

It is obviously our intention to have this interface be used with programs other than just this test aid. If you want to make things easy, you can use the Soft Controller I and II programs developed by the author.

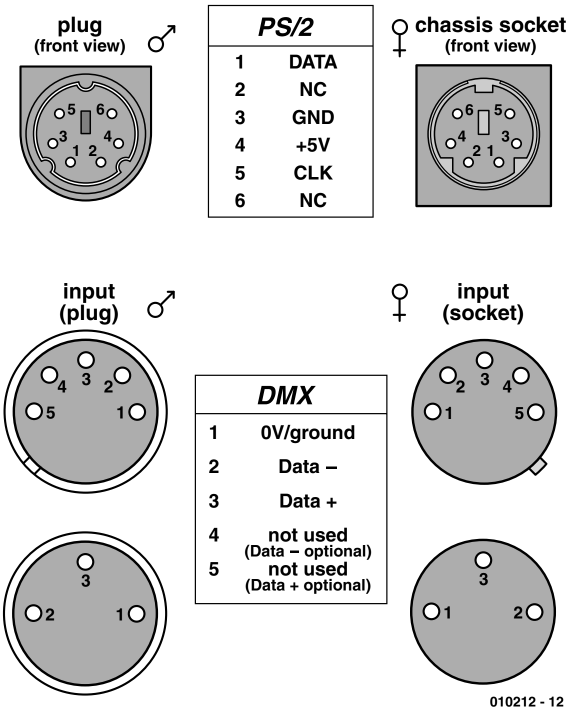


Figure 3. Using this connector information, you can ensure that the interconnecting cable is properly attached to the circuit board.

These have already been briefly discussed in the article describing the MIDI/DMX interface, with which they are also compatible. For more information, you can send an e-mail to the author. Still, as already mentioned at the beginning of this article, the main advantage of this circuit is that it can be easily programmed, even under Windows.

While most commercially available DMX interfaces for PCs cannot be used without a lot of control programs and drivers, our interface uses a protocol that is available on every PC: the Centronics protocol.

The interface acts like a printer by emulating the appropriate commands. This means that in order to have the interface execute a command, all you have to do is to instruct the PC to print a series of characters. Printing functionality is included in various flavours of programming languages, such as Delphi, C++ Builder, Visual Basic and the like, which means that the biggest problem is already solved.

If you program under DOS, you can simply use the Basic LPRINT instructions to communicate with the interface.

As already noted, this interface is not limited to use with PCs. Any device with a Centronics port is by definition compatible with the interface.

With regards to programming, here we limit ourselves to describing the method that must be used to send commands to the interface. For readers who program under Windows, we have developed a small DLL library that can be found on the floppy disk (as well as on our Internet site for free downloading). This means that you do not have to trouble yourself with learning how to use the Windows API (Application Programming Interface).

Programming the interface is actually very simple. The interface recognises a certain number of commands, each of which consists of a letter that may be followed one or more parameters, depending on the particular command. To allow the interface to recognise the start of a command, the ESC code ('27' in ASCII or '1B' in hexadecimal) is used as an identification code.

As an example, let's see which bytes you have to send to the 'printer' to set DMX channel 18 to a value of 155. For this purpose we use the command 'd', followed by the channel number (which can range from 1 to 256, but here we use 0 to 255 to allow it to fit into a single byte), and then followed by the DMX value to be sent.

The bytes to be sent to the parallel port thus appear as follows (the hexadecimal values are shown in parentheses):

```
ESC  (1Bh)
d (64h)
17(11h)
155  (9Bh)
```

Another example: to reinitialise the interface, all you have to do is send the command 'Z'. The byte sequence is thus:

```
ESC  (1Bh)
Z (5Ah)
```

Knowing this, all you have to do to create your own 'program of the century' is to write your own series of bytes and send it to the interface.

Table 1 summarises all the commands recognised by the interface.

(010202-1)

Disk contents (010212-11)

8515def.inc	file containing the definitions of the 8515 registers
defio.inc	file containing the descriptions of the input and output pins
Lptdmx.asm	assembler file for the LPT/DMX interface (in French)
Lptdmx.hex	hex file for the LPT/DMX interface
LPT_DMX_LIB	folder containing the DLL library for controlling the interface under Windows, along with the file 'readme' (English)
LPTDMX_TESTER.EXE	test program for the interface
LPTDMX_TESTER_E.DOC	English-language user's guide for the test program
LPTDMX_TESTER_F.DOC	French-language user's guide for the test program